Timely Notes and Supplementary Information for

C-terp

Version L 2.14
(Lattice Variant)

GETTING STARTED

The install program, described in the GETTING STARTED section of the manual, has been extended to include setting the BIOS switch (this directs all screen I/O through the BIOS) which allows C-terp to be run on machines that are BIOS compatible with the IBM PC.

The install program also allows the cursor sizes to be set.

DEBUGGING

Within the Debugger, the Display command is now case sensitive. If a 'd' command is given the behavior is the same as before. If a 'D' command is given, the screen is flipped to its execution state before the expression is evaluated so that if it is doing screen I/O the results can be seen on the flipped screen.

LANGUAGE EXTENSIONS

In the section LANGUAGE EXTENSIONS the example of the ESC character should be '\x1B' rather than '\x27'.

EDITING

Long Lines

Starting with version 2.13, the editor supports horizontal scrolling implying that you may directly edit information that would normally lie beyond the right edge of the screen. Subsection 'Long Lines' in Section 'THE EDITOR' is now obsolete.

New Editor Commands

The following editor commands have been added since the document was written:

Alt B Bring Back the old line (undo the editing changes made to this line). Once the cursor has moved off a given line the changes have become permanent.

Alt D Delete the current range of marked lines.

Alt E Erase to End of line.

Alt O Print Out the marked set of lines (to the printer) in the current file or, if there is no set of marked lines in the current file, print out the entire file. You may

interrupt the printing with a Ctrl-Break. The printout will contain the line number. Wrap-around will take place if the line is too long as defined by a parameter in the config.c file.

Modified Editor commands

The Alt I (Interchange command) now provides one additional letter, 'l', in its list of alternatives which now reads: (y/n/q/l). The 'l' stands for 'last'. It means that this is the last of a sequence of interchanges so that the editor will make the change but then go no further.

Prepending a Line

To prepend a line to a file while in the editor (i.e., to add a line before line 1) place the cursor at line 1 column 0 (you can use Ctrl Home for this purpose) and do a Line Split (Alt-S).

LIBRARY

Complete Pointer Checking

Since the manual was written, a new check level, check(2) has been introduced (see the function: check(level) in the section entitled THE LIBRARY). This is now the default checking level. Check level 2 will check each pointer for residing within (a) one of the automatic areas, (b) the static area, (c) one of the malloc'ed areas (includes calloc and realloc) or (d) one of the

external data areas. This is in contrast to high-water mark, low-water mark pointer checking of check level 1 and the almost no checking of check level 0.

There is a slight performance degradation in the use of check(2) which is unnoticable for small programs which do not use many malloc's.

Additional Library Functions

By popular demand we have added the following library functions from your compiler's library; these are not described in the manual.

gets
puts

LINKING

Complete Library

The file ALL.BAT is a sample batch file for including additional (actually, just about all) members of your compiler's library into C-terp. The file ALL.AUG contains the names of the additional functions to be added.

Editor Customization

Several files have been added to the distribution diskette(s) for the purpose of customizing C-terp (particularly the editor). These are:

config.h
config.c
cf_test.c

config.c is the module to modify if you want to reconfigure C-terp. config.h is an include file and theoretically should not be modified. cf_test.c is a small test program described below.

config.c contains a table associating character sequences with editor actions. Character sequences of length up to three may be employed. Certain parameters and flags which affect the behavior of the editor and other portions of C-terp can also be set. You may also modify the editor help screen to correspond with the changes you have made. Please refer to the files config.h and config.c for more information.

You may temporarily modify the configuration tables via C-terp and in this way test anticipated changes interactively. Program cf_test.c is an example of such a program. By running it you will notice that the editor suddenly behaves differently (for the course of the C-terp session). Feel free to alter cf_test.c to change the editor in other ways before making them permanent.

To customize C-terp, modify and then compile config.c using the large model (see LINKING TO EXTERNALS in your manual) producing object module config.obj. Then issue a link command similar to the link command described in your manual but include config.obj alongside tblxn.obj. Thus, the link command should look like:

LINK ... tblxn+config, ctnew, ... etc.

Saving the Graphics Screen

We have introduced the ability for graphics users to customize the saving and restoring of their

screen (across a debugging break point).

Before C-terp enters debugging mode it calls

```
scr_save(1);
```

It is the responsibility of scr_save to save the current screen and mode and set the screen to 80 X 25 character mode. When C-terp exits from debugging in order to continue on with the program execution or when a capital letter debug command is issued ('D', 'S' or 'N')

```
scr_save(-1);
```

is called. With an argument of -1, scr_save should restore the old screen and mode. Finally,

```
scr_save(0);
```

is called once at C-terp start and is used for initialization.

scr_save is housed in a file called scrsave.c. This can be found on the distribution diskettes.

To add your own screen saving to C-terp, modify the scrsave.c file and compile it using the large model (see LINKING TO EXTERNALS in your manual) producing object module scrsave.obj. Then issue a link command similar to the link command described in your manual but include scrsave.obj alongside tblxn.obj. Thus, the link command should look like:

```
LINK ... tblxn+scrsave, ctnew, ... etc.
```

Too many segments

If you are using the DOS 3.0 linker you may get a "Too many segments" error message when making a new version of C-terp (see LINKING TO EXTERNALS in the manual). The default number of segments is

128. To obtain additional segments use the /SE switch. For example, to obtain 200 segments use the command:

LINK ... tblxn /SE:200, ctnew, ...

The maximum number of segments is 1024.

If you are using Version 2.30 of the Linker use 'X' in place of 'SE'.

Lattice version dependencies

The file c.obj furnished on the distribution diskette corresponds with version 2.15 of the Lattice compiler. If you are using the Lattice 2.14 compiler please use c214.obj in place of c.obj when you are making a new version of C-terp (see Section LINKING TO EXTERNALS).

For the purpose of showing how c.asm is modified for C-terp, the source for c.asm has been included under the names c214.asm and c215.asm.

CAUTION: You may not link in libraries with versions of Lattice C earlier than Version 2.14

The Large Model Switch

Instructions within the document (section LINKING TO EXTERNALS) indicate that you must use the -ml and -s switches when compiling a module to be linked with C-terp. If you are using the Lattice's lc.com, then use:

lc -mls ModuleName

Note that the -mls goes before the module name, not after.